PROGRAMMING

MERIT BADGE POWERPOINT

© 2018 - 2020 ROBERT BAKER

WHAT IS PROGRAMMING

WHAT IS PROGRAMMING?

- Programming is the act of inserting instructions into a computer or machine to be followed.
- There are many different career fields involving the programming of computers; each utilizing different languages, techniques, and systems.
- We are only going to cover a few of the different aspects of programming during this Merit Badge, but there are so many more.

SAFETY

- Normally programming normally involves computers, which use electricity. It is important to make sure all power-cords are not frayed, and too keep liquids far away to prevent electric shock.
- RSI – Repetitive Stress Injury
  - Caused by typing for long periods of time and can cause pain in the wrists and hands
  - How can RSI be prevented?

SAFETY

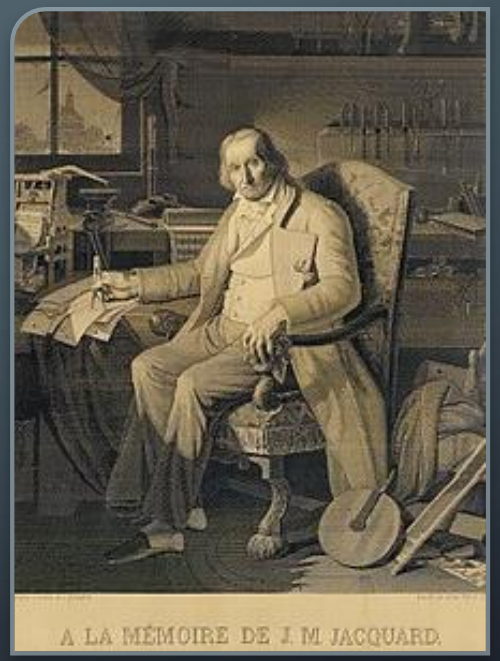
- Eye Strain can be caused by using computer screens for extended periods of time.
  - How can eye strain be prevented?

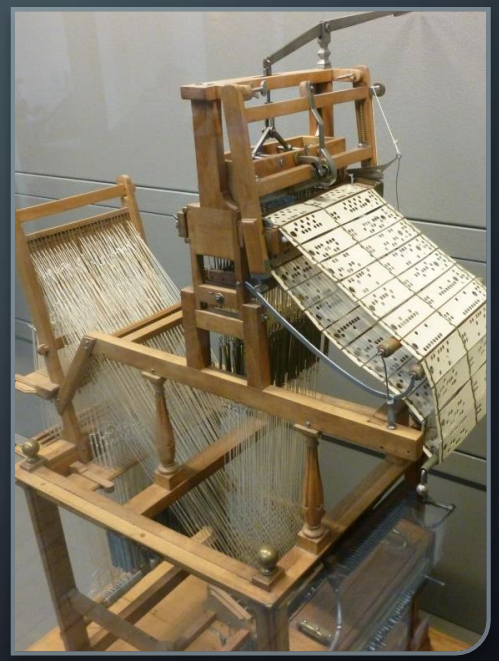
BEFORE COMPUTERS

Before the modern electrical computer, mechanical devices used in factories were the first machines to be programmed.

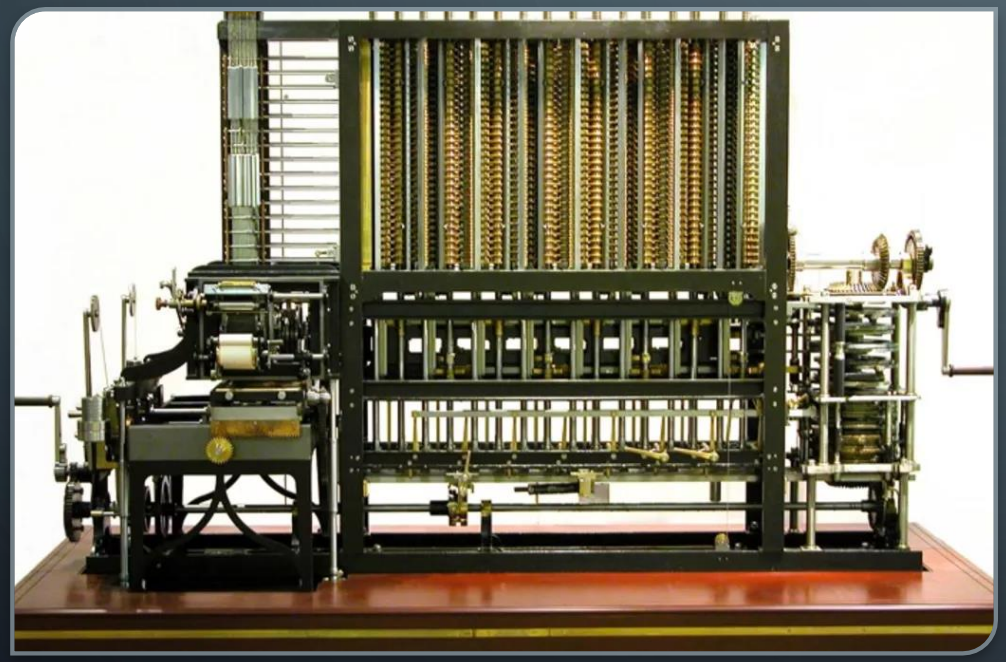
An example is the Joseph Jacquard Loom (1804) which used hole-punched cards to “program” patterns into fabric.

The picture on the left is the loom.
The picture on the right is a portrait of Jacquard was woven in silk on a Jacquard loom and required 24,000 punched cards to create (1839). One of these portraits in the possession of Charles Babbage inspired him in using perforated cards in his Difference Engine.





BEFORE COMPUTERS



Ada Lovelace, the first programmer, theorized how to program Babbage's Machines.

Charles Babbage in 1823 started work on his Difference Engine. It was programmed using punch cards and could do simple calculations to 31 digits. Do to high costs, it was not built until 1991, well after his death. It weighed 15 tons and was 8 ft tall.

It used human-power to turn the gears and cranks and output the result using wheels with digits painted on.

Fun Fact: The gear technology didn't exist to build his machine, so Babbage invented new ways of cutting gears. This incidentally advanced machinery and factories during the end industrial revolution (1760-1840).

BEFORE COMPUTERS

In 1885, Herman Hollerith designed the “Electric Tabulating System”, a machine designed to take on the 1890’s Census. It was an early Scantron-like machine using punch cards.

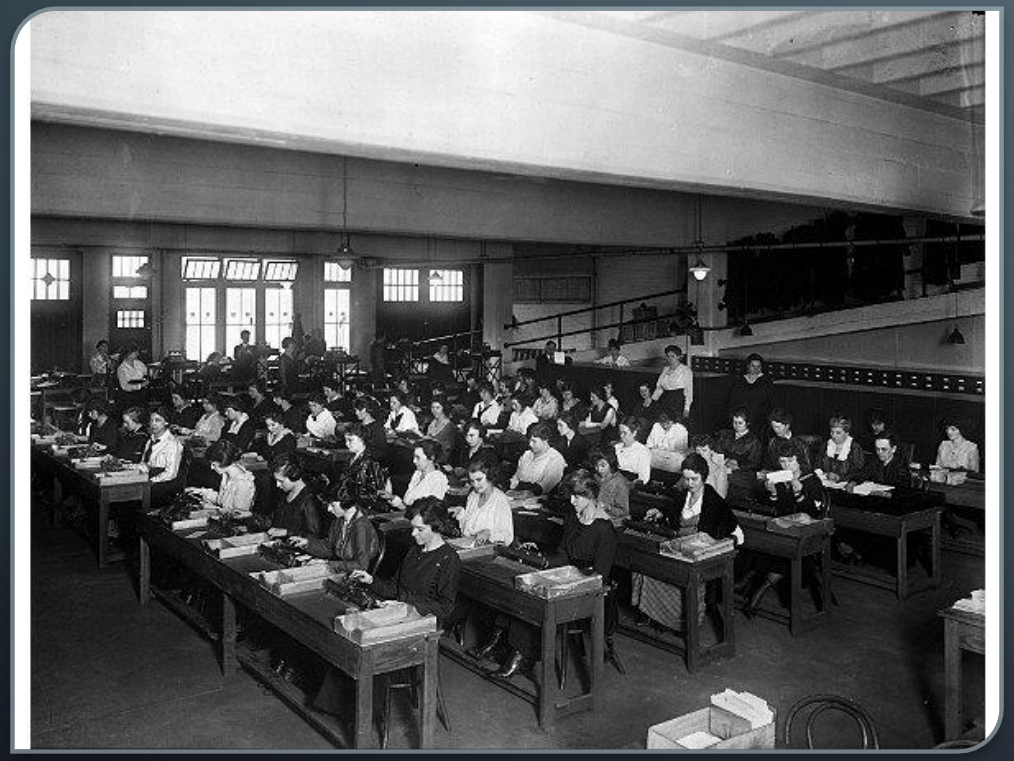
The 1880’s Census took 7 years to count, so due to the growing population, the 1890’s and 1900’s Censuses would have taken more than 10 years. This would not be good.

With his machine, the 1890’s Census only took 6 weeks rather than 10 years. This proved computers were a viable solution to many previously impossible problems.

FIRST COMPUTERS

WHAT DID THE FIRST COMPUTERS LOOK LIKE?

FIRST COMPUTERS



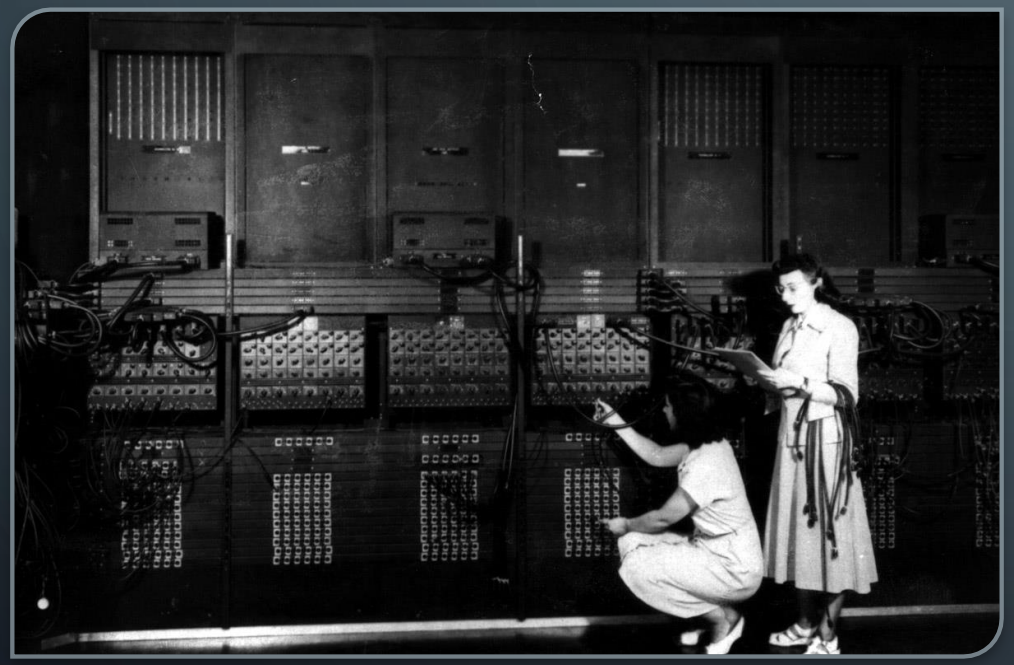
A “kilogirl” was a unit of measurement equaling 1000 hours of computing labor

A “Computer” used to be a job description, not an electronic machine

Women almost exclusively filled these positions.

Large agencies would have “Computer Rooms” with many ladies doing calculations by hand

EARLY COMPUTERS



ENIAC 1946 – What do you notice about this photo

ENIAC – Electronic Numerical Integrator And Computer (1946)

- First general-purpose computer
  - Used Base-10 instead of Binary (Base-2)
- They used Vacuum Tubes and Mechanical Switches
- Used to calculate firing-tables for the military.

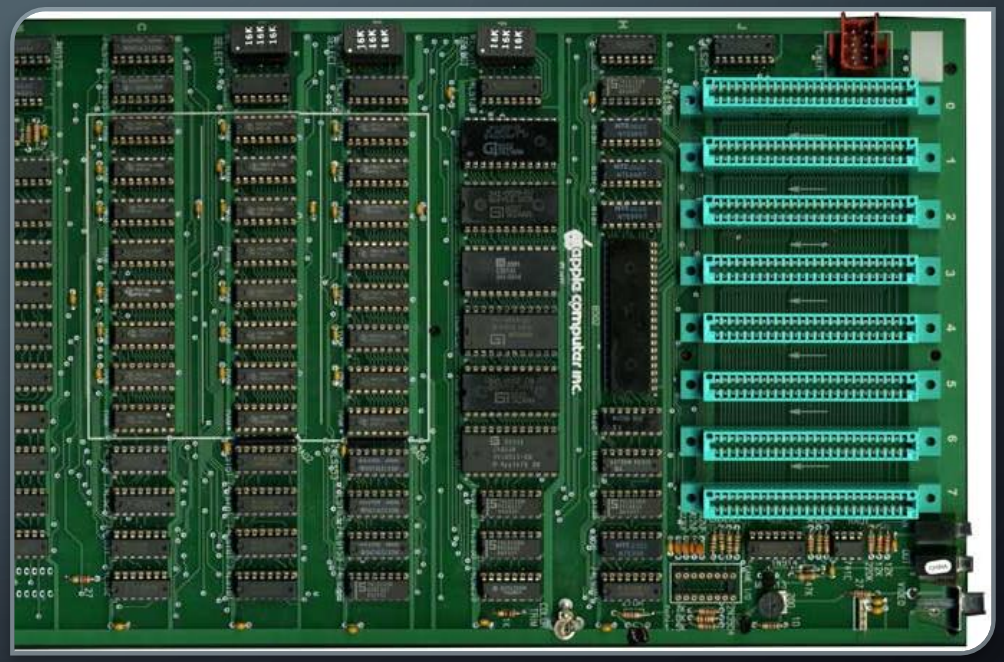
UNIVAC – UNIVersal Automatic Computer (1951)

- First commercial computer
- Brought computers into the public eye after it correctly predicted the “total-upset, landslide”, 1952 Presidential Election.

PRE-MODERN COMPUTERS

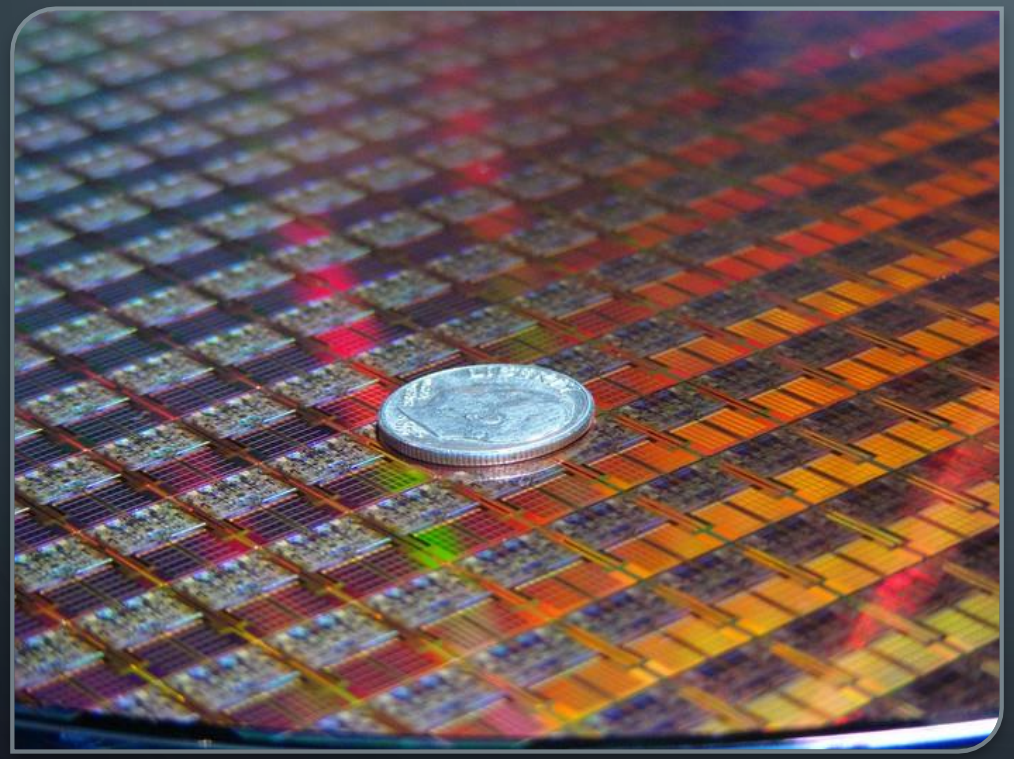
After Vacuum Tube and Mechanical Switched computers, Integrated Circuits (ICs) allowed computers to get much smaller. Computers when from the Size of buildings to the size of desks.

This also allowed more powerful computers to be built because less space was needed.



This is an Apple2 motherboard.
All the black chips make the CPU.
Each one is about 1” wide.

MODERN COMPUTERS



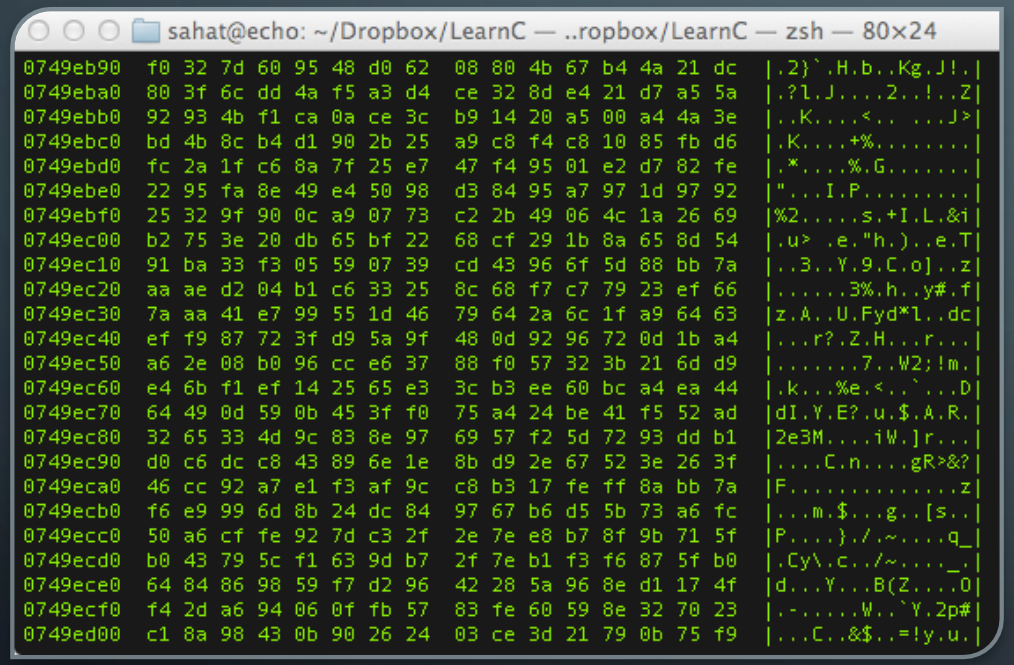
Each “switch” in these chips are 10nm wide

The Microprocessor allowed computers to go from the size of desks to the size of a dime!

Each small square in this picture is a computer!

This allowed use to make computers even more powerful and allow us to use even more powerful language features.

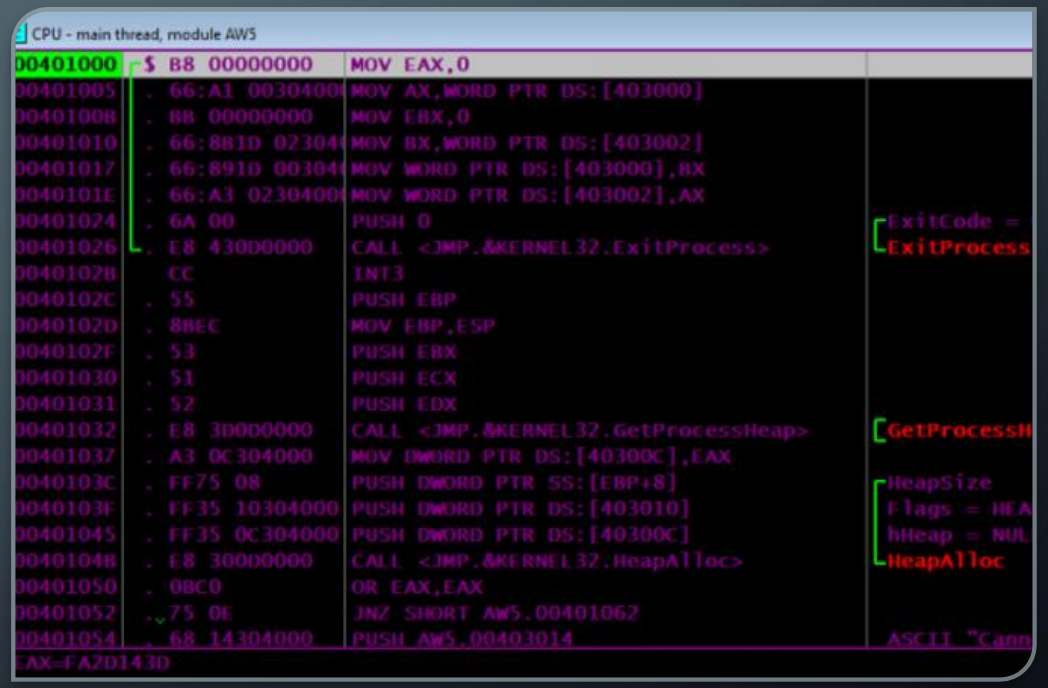
HISTORY OF PROGRAMMING



- What was the first programming language?
  - Binary / Machine Language (ML)
- Binary / ML is really hard to read, but it can be done.
- Early computers used switches and cables to accomplish this.
- It is insanely fast, only limited by hardware speed.
- All programming languages end up as Binary / ML at some point during execution.

HISTORY OF PROGRAMMING

- Next came Assembly Language (ASM)
- Slightly easier to read than Binary / ML
- Still very fast because it maps back to Binary / ML
- Very few people 'need' to program is ASM
- There is a different Assembly Language for each CPU design, so it is not portable code.
  - Why is portable code good?



HISTORY OF PROGRAMMING

Next-Generation Languages came around the 1950's.

They allowed:

- Code portability between different systems
- Easier to write, read and debug code
- Allowed for new concepts (i.e. functions, classes, objects, OOP)
- Explored new fields (i.e. science, math, computer science, data science, business)

The first big languages were… (in order of creation)

FORTRAN, LISP, COBOL, BASIC and Pascal

PROGRAMMING NOW

How many languages do you recognize?

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL
SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust
F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM
RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

PROGRAMMING NOW

Why are the languages grouped into colors?

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

The Green Languages are General Programing Languages
The Purple Languages are Scripting Languages
The Red Languages are Markup Languages
The Blue Languages are Declarative Languages
The Orange Languages are Assembly Languages

Different types of languages have different purposes.
It is important to match the type of work to the correct language to insure the best results.

PROGRAMMING LANGUAGES

Here are a few languages and the problems they try to tackle…

C++ – General Purpose, High Performance | ex. Game Engines, Desktop Apps (Adobe Photoshop, Chrome)
C – General Purpose, High Performance, Light Weight | ex. Linux OS, macOS, Integrated Circuits, Drivers
Java – General Purpose, Multiplatform | ex. Minecraft, Server Apps, Android Apps
C# – General Purpose, Windows Platform | ex. Unity Games, Server Apps, StackOverflow
Swift – General Purpose, iOS & macOS | ex. most apps for iPhones and macOS (replaced Objective-C)
SQL – Database Communication
JavaScript – General Web Scripting | ex. Interactive webpages, webpages that can run dynamic code
HTML – Webpage Design, Layout and Markup
CSS – Webpage Styling, Coloring, Fonts and Positioning
PHP – Web Server Code | ex. Backend Web Dev., Web Content Management Systems (i.e. WordPress)
TypeScript – Stricter Superset of JS that transpiles into JS |ex. Large JavaScript Apps
XML – Human and Machine readable file format for data sharing between apps

PROGRAMMING EXAMPLES

Hello World

C++

```
#include <iostream>
int main(int argc, char *argv[])
{
     char myString[] = “Hello World!”;
     std::cout << myString << std::endl;
     return 0;
}
```

Java

```
class HelloWorld {
   private String myString = "Hello World!";
   public static void main(String args[]) {
      System.out.println(myString);
   }
}
```

Notice how different languages can look very different even when they are doing the same task.
Notice also how the bracing (i.e. “{}”) style is different between languages.

PROGRAMMING EXAMPLES

Hello World

C#

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
   class HelloWorld
   {
      String myString = "Hello, world!";
      static void Main(string[] args)
      {
         Console.WriteLine(myString);
      }
   }
}
```

X86 Assembly

```
.486
   .model flat, stdcall
   .stack 100h
   option casemap :none

   ExitProcess PROTO Near32 stdcall, dwExitCode:dword
   putch PROTO Near32 stdcall, bChar:byte;
.data
   strMyString byte "Hello World",0
.code
main PROC
   mov ecx, LENGTHOF strMyString
   mov esi, OFFSET strMyString
L1:
   invoke putch, byte PTR esi
   inc esi
   loop L1
   invoke ExitProcess,0
main ENDP
END main
```

PROGRAMMING EXAMPLES

Hello World

JavaScript

```javascript
myString = "Hello World!";
console.log(myString);
```

Python

```python
myString = 'Hello World!'
print(myString)
```

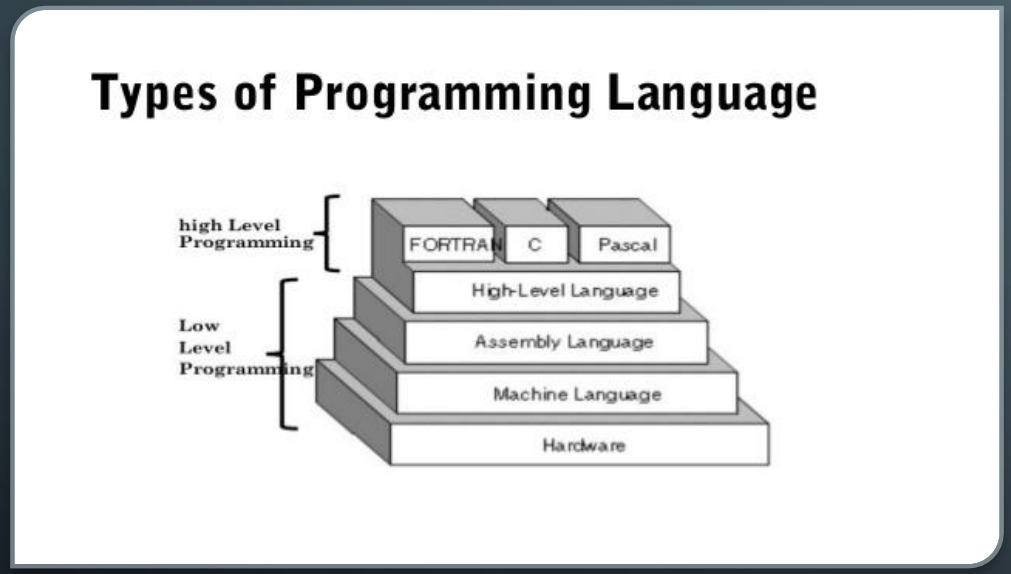
Notice how different languages can look very different even when they are doing the same task.

PROGRAMMING LANGUAGE TYPES

Languages can be split into a three different levels..

- High-Level (ex. Python, Ruby, JavaScript, Java, SQL)
- C-Level (ex. C, C++, Rust)
- Low-Level (x86 Assembly, Machine Language)

PROGRAMMING LANGUAGE TYPES



Note: Java, Python, etc. are one level higher than FORTAN, C and PASCAL

Why would you use a High-Level, Low-Level or C-Level language?

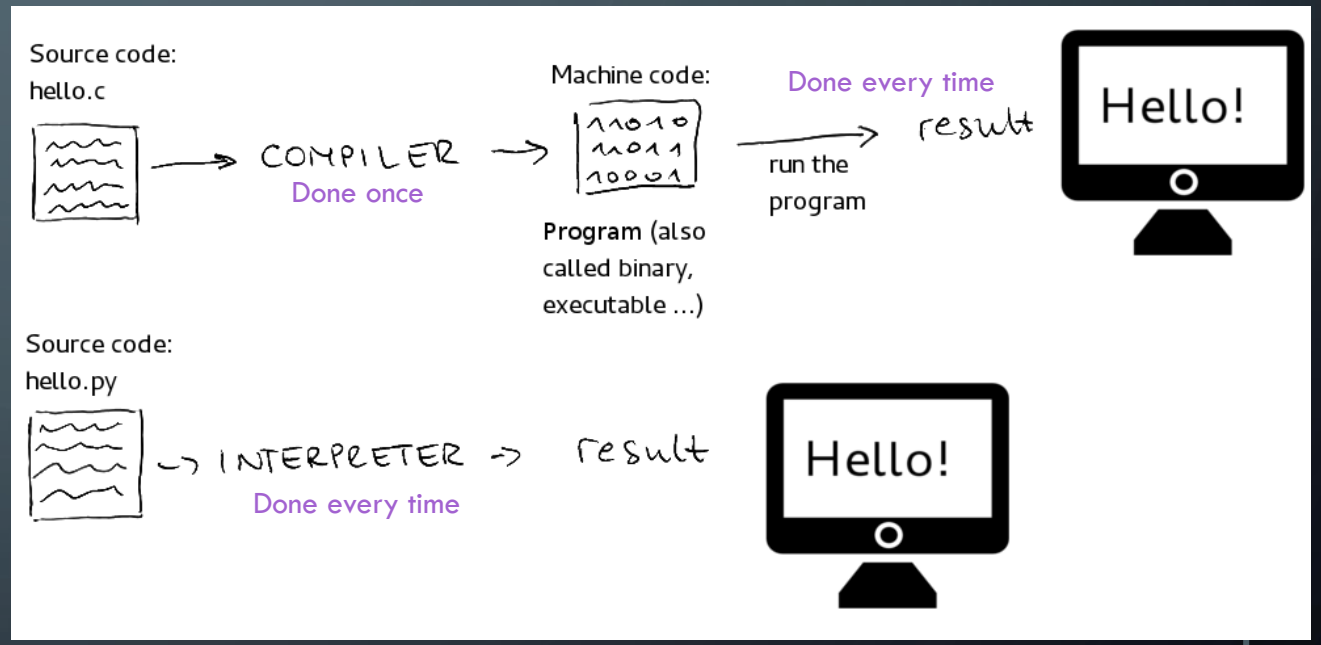
- Low Level
  - Pros: Fast Execution, No Overhead, Single Platform, Compiled
  - Cons: Hard to read, write, debug, and maintain
  - Examples: ML, MASM, TASM, NASM, MIPS
- High-Level
  - Pros: Easier to read, write, debug, and maintain, Multi-Platform, Compiled or Interpreted
  - Cons: Slower than Low-Level, not as much control over hardware
  - Examples: Python, Ladder-Logic, JavaScript, Java, SQL
- C-Level
  - Best of both worlds, Compiled
  - Good control over hardware with ease of writing.
  - Examples: C, C++, Rust, FORTRAN, PASCAL

PROGRAMMING LANGUAGE TYPES

This photo illiterates the difference between compiled and interpreted languages…

Use your computers to make a list of 3 compiled languages and 3 interpreted languages.

Where would you use a compiled languages vs an interpreted language?



PROGRAMMED DEVICES

Our lives are filled with so many programmed devices, you many not even notice…

What are somethings around your house that are programmed?

- Smart TVs, Smart Door Bells
- Xbox, PlayStation, Wii, Ms. Pacman
- Microwave, Wi-Fi Router (these two are the same thing)
- Etc..

What language do you think these were programmed in?

INTELLECTUAL PROPERTY (IP)

What are the four types of IP?

1. Copyright
2. Patent
3. Trademark
4. Trade Secret

Open your computers and go online. Get the definitions of all four of these types of IP.

INTELLECTUAL PROPERTY (IP)

What is …

1. Copyright – protects a particular expression of an idea that the author created (i.e. PowerPoints, Game Art, Specific Code)
2. Patent – protects useful innovative processes or methods, machines, manufactured items, or "compositions of matter" (i.e. a new and revolutionary math algorithm used in an app)
3. Trademark – protects a word, phrase, symbol or sound that identifies and distinguishes the source of a particular product or service (i.e. Windows Logo, "Your mattress is freeee", etc.)
4. Trade Secret – protects valuable information be not disclosing it to anyone, enforced by a contract called a NDA (i.e. what info Facebook collects)

OWNING VS LICENSING

Do I own a copy of PowerPoint?

Do I own a copy of Google Chrome?

Do I own a copy of an App I built?

What is the difference between owning and licensing?

- Owning means you can do what ever you want to the software. Most people do not own software.
- Licensing is where you “buy or get permission” to use the software, often subscription based.

LICENSE TYPES EXPLAINED

Use your computers to research the following key terms…

- Open-Source –
- Closed-Source –
- Freeware –
- Shareware –
- Demo –
- Public Domain –

LICENSE TYPES EXPLAINED

Use your computers to research the following key terms…

- Open-Source – the code is exposed to the public and can be modified or distributed, may be limits or restrictions (doesn't mean free).
- Closed-Source – the code is NOT exposed to the public and cannot be edited or distributed (doesn't mean free).
- Freeware – 100% free to use, not necessarily free to be modified or distributed.
- Shareware – free to download and use, but asked for donations (i.e. Ad-Block). Not free to modify or distribute.
- Demo – A free trial version of the program, may not have all the features enabled. Not free to modify or distribute.
- Pubic Domain – There is absolutely no ownership such as copyright, trademark, or patent. Software in the public domain can be modified, distributed, or sold even without any attribution by anyone.

CAREERS IN PROGRAMMING

What are some careers you have heard of in a programming field?

- Computer Scientist
- Mobile App Developer (dev)
- Desktop App Dev
- UI / UX Engineer
- Software Engineer
- Game Engine Dev
- Gameplay Dev
- Database Engineer
- Hardware Engineer
- Computer Engineer
- Sysadmin
- Hacker / Pen-Tester
- Web Dev (frontend and backend)

REVIEW / LUNCH / PROJECT TIME

Room Number	Language	Rotation 1	Rotation 2	Rotation 3
SM346	C++	Group 1	Group 2	Group 3
SM348	C#	Group 3	Group 1	Group 2
SM202	HTML/CSS	Group 2	Group 3	Group 1

SM346 and SM348 are on this floor
SM202 is one floor down

Before lunch, your knowledge will be reviewed by pairs of Merit Badge Councilors and TA's.
Let's head outside and make lines of 8-10 Scouts per line. Bring everything with you.

Once you have been reviewed successfully, you may eat lunch.
There will be some booths you can visit during lunch to learn about different CS Clubs in college.